第9卷 第2期
2017年4月
土木建筑工程信息技术
Journal of Information Technology in Civil Engineering and Architecture
Vol. 9 No. 2
Apr. 2017

建筑信息模型的Web端重建与三维交互方法研究

李昌华 张晗玥 周方晓

(西安建筑科技大学信息与控制工程学院,西安 710055)

【摘 要】在BIM技术的背景下,针对现有的BIM服务器多为C/S架构,对电脑软硬件有较高要求、无法跨平台等问题,实现BIM模型在Web端的重建,并实现了对模型的拾取,获取其属性。其方法主要分为两部分,第一部分实现BIM模型从Revit软件中的导出,第二部实现对其导出的模型信息在Web端的三维交互,具有良好的用户体验。

【关键词】建筑信息模型; WebGL; 三维交互
【中图分类号】TU17 【文献标识码】A 【文章编号】1674-7461(2017)02-0047-05
【DOI】10.16670/j.cnki.cn11-5823/tu.2017.02.08

引言

建筑信息模型(Building Information Modeling)是近年来兴起的新概念,在建筑的设计、施工、运维的全生命周期中应用价值极高,已成为建筑业的新趋势[1]。目前主流的BIM服务器多为C/S架构,而C/S架构的桌面应用程序对电脑软硬件和软件使用者都有一定的要求[2]。随着互联网的发展和移动终端的普及,在Web端展示BIM模型已成为BIM从业人员的潜在需求,具有极高应用价值。因此,对于建筑信息模型在Web端重建与三维交互的研究是十分必要的。

本文基于在Revit平台下建立好的模型,使用C#语言,通过对Revit进行二次开发,提取出建筑模型几何数据及属性信息。利用WebGL技术构建三维交互平台并加载三维模型,利用计算机图形学技术实现鼠标和网页三维场景中的模型进行互动。

1 Revit中的BIM数据导出

Revit Architecture软件是Autodesk公司专门对建筑行业开发的BIM工具,在建筑领域应用非常广泛[3]。Revit平台是开放的,它具有强大的二次开发端口,提取模型信息可以在Revit平台下通过Revit API相关属性来实现。对Revit模型信息的提取,其中包括模型几何数据,材质、纹理等属性数据。由于本技术应用前提是在三维视图中,因此程序开始时要判断doc.ActiveView是否为View3D对象,当不在三维视图下弹出警告框。RevitAPI提供了专门的导出类CustomExporter,通过自定义一个导出类CMyExporter,继承IExportContext类,实现接口里面所有的方法,IExportContext导出类中的Onmaterial()方法,可以获取到材质相关信息,插件还调用了OnPolymesh、OnElementBegin、OnElementEnd等函数接口,用于获取Revit模型的节点属性,例如坐标点、材质信息、贴图数据等。

该数据导出插件开发步骤如图1所示。

依据上述步骤,可以在Revit软件上开发自动提取建筑信息模型数据的功能插件。应用具体操作如下:在Visual Studio 2013平台上,引用Revit接

【基金项目】国家自然科学基金项目“基于空间句法的非精确图匹配方法研究及应用”(项目编号:61373112);陕西省自然科学基金项目“基于建筑空间关系的BIM模型分类与检索方法研究”(项目编号:2016JM6078)
【作者简介】李昌华(1963-),男,工学博士学位,西安建筑科技大学教授、博士生导师,主要研究领域为图形图像处理,模式识别,数学建模等;张晗玥(1992-),女,硕士研究生,主要研究方向:建筑业信息化;周方晓(1971-),男,工学博士学位,西安建筑科技大学讲师,主要研究领域为数字建模等。

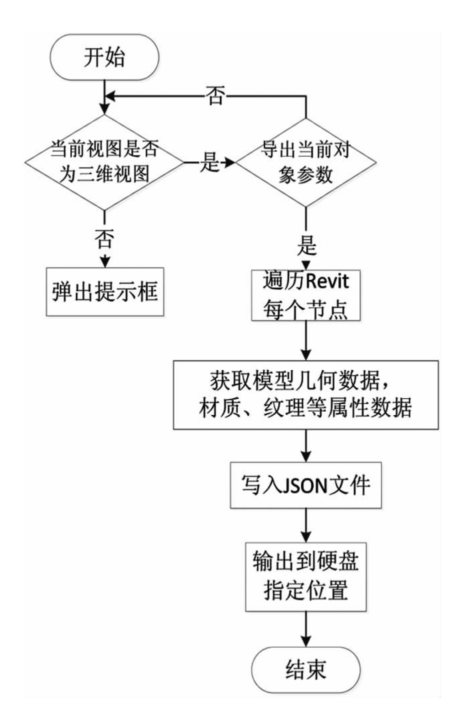


图 1　数据导出插件开发步骤

口定义文件 Revit-API. dll 和 RevitAPIUI. dll 以及部分的 Revit API 提供的命名空间，使用 C#语言按以上流程完成插件的开发。然后再 Revit 平台下选取附加模块调用该插件即可。

在数据导出插件的开发过程中，需要解决以下几个关键问题。

（1）获取几何数据

例如墙的几何数据主要是一个由面和边组成的几何实体，由以下步骤可获取：

1）通过墙类型的几何属性 Geometry 获取 GeometryElement 的实例。这个实例里面存储了所有相关的几何对象，如实体、线等；

2）遍历 GeometryElement 实例来得到一个几何实体 solid，这个几何实体 solid 的 Faces 和 Edges 属性里面包含了所有的几何面和边；

3）遍历 Faces 属性得到所有几何面；

4）遍历 Edges 属性得到所有的几何边。

（2）获取材质信息

获取模型材质信息，主要使用 Revit 中所提供的 API，IExportContex 导出类中的 OnMaterial( ) 方法，Material 对象中包含材质名，颜色，填充图形等信息。核心代码如下：

```
//利用 Material 的 AppearanceAssetId 属性得到 AppearanceAssetId
ElementId appearanceId = material. AppearanceAssetId;
//通过上面取得的 AppearanceAssetId 取得 AppearanceAssetElement
AppearanceAssetElement appearanceElem = document. GetElement ( appearanceId ) as AppearanceAssetElement;
//获得 Asset
Asset theAsset = appearanceElem. GetRenderingAsset( );
```

2　BIM 模型的 Web 端重建及交互

通过以上插件，已将 Revit 中构建的模型转换成 JOSN 格式文件，JSON 是可被 WebGL 解析的通用数据格式。本文借助 WebGL 的第三方引擎 Three. js 来实现其功能。

2. 1　模型载入

Three. js 需要依托网页才能发挥作用，系统需要建立在 HTML 结构之上，获取最新 Three. js 包之后，在 HTML 的 head 里面的 < script > 标签里面嵌入 Three. js 库文件，首先初始化一个渲染器对象 render，然后创建一个场景，在 Three. js 中场景，场景用于容纳所有的其他对象，再往场景中添加一个相机，相机定义了我们从哪里观察场景，WebGL 中的相机有两种：正投影相机和透视相机，透视相机类似于真实世界中的相机，近处物体比例大，远处物体比例小，本文使用透视相机，最后再调用渲染器的 render( ) 方法，来处理场景和相机。Three. js 的基本渲染结构如图 2 所示。

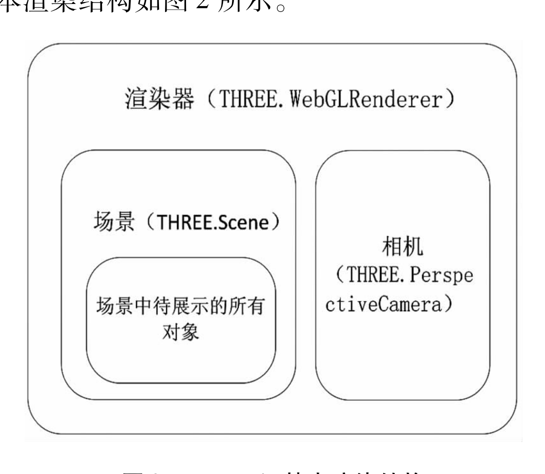


图 2　Three. js 基本渲染结构

Three. js 的编程原理较为简单，一般流程为：

1）对渲染器、场景三维对象、相机进行初始化设置；

2）将三维对象添加到场景中；

3）将场景和相机添加到渲染器中；

4）用渲染器将其在屏幕上渲染出来。

在 Three. js 中，把模型加载到浏览器的过程如图 3 所示：

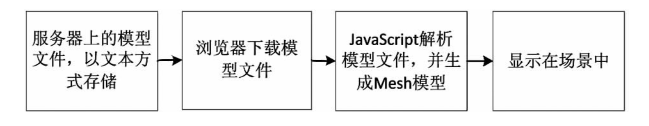


图 3　模型加载过程

服务器上的模型文件是以文本方式存储，存储了模型的顶点、材质等信息，第二步是浏览器下载文本文件，使用 javascript 的异步请求来实现，然后 JavaScript 解析文本文件，生成一个 geometry 再生成 Mesh 模型，最后将其加入到场景中。在 Three. js 中 THREE. Mesh 用来表示网格模型，它的构造函数是 THREE. Mesh = function（geometry，material），其中第一个参数是一个 THREE. Geometry 类型的对象，定义了顶点和顶点之间的连接关系，第二个参数定义了模型材质属性。

导出后的 JSON 文件在前文中有详细的分析，Three. js 框架中内置了 THREE. JSONLoader（）类，该类可以对 JSON 格式的模型进行解析，并载入到浏览器中，将前文所导出的 JSON 格式 Revit 模型通过该类可载入到浏览器，载入时需要获取 Revit 模型的几何信息，包括顶点数据，面数据，及材质信息，包括所导出的模型材质、属性。载入的模型放入 Three. js 的场景中，GPU 会读取顶点信息，并通过顶点着色器中来处理每个顶点，完成模型绘制。

2. 2　模型拾取

在前文已获取模型 JSON 文件的基础上，为实现用户与建筑信息模型在 Web 端的三维交互，就要使用 WebGL 技术，然而 WebGL 只是三维图形库，其中并没有内置任何点击检测的方法，这需要经过一系列的数学变化来自己构建。其中最为重要的三维交互之一就是处理并找出鼠标指针位于哪一页面元素之上，这种交互式的可视化程序可以大大增强用户体验。

首先要实现鼠标的点击检测，第一步要获得射线的矢量。鼠标点击屏幕上 p'点时，对应于投影空间中的点为 p，在视图空间中 p 的对应点是 v，在视图空间中的原点和 v 两点相连形成一条射线。最后还需要将射线的坐标转换为世界坐标。

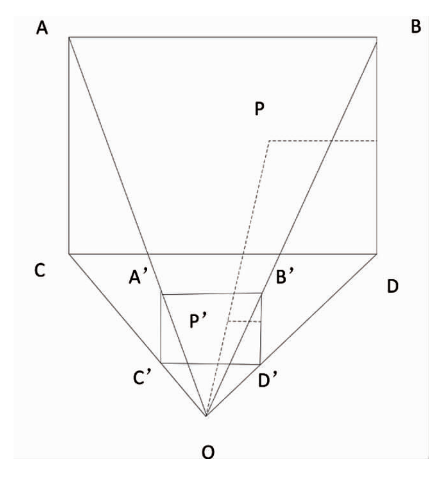


图 4　模型拾取射线相交法例图

（1）将鼠标点击屏幕的点坐标变换为投影空间中近裁剪面（A'，B'，C'，D'）中的坐标，定 Z 轴深度为 0. 5。使用数学的解析几何运算，得到以下变换公式：

$$\begin{cases} p_x = (x - screenWidth/2)/screenWidth * 2 \\ p_y = -(y - screenHeigth/2) screenHeight * 2 \\ p_z = 0.5 \end{cases} \quad (1)$$

（2）为得到在视图空间中的坐标，对上一步中得到的投影空间中的点坐标进行矩阵变换，将投影空间中原点设为近剪裁面（A'，B'，C'，D'）的中心。

$$pMatrix = \begin{bmatrix} w & 0 & 0 & 0 \\ 0 & h & 0 & 0 \\ 0 & 0 & Q & 1 \\ 0 & 0 & -QZn & 0 \end{bmatrix} = \begin{bmatrix} m11 & 0 & 0 & 0 \\ 0 & m22 & 0 & 0 \\ 0 & 0 & m33 & 1 \\ 0 & 0 & m43 & 0 \end{bmatrix} \quad (2)$$

$$\begin{cases} w = cot(ang1/2) * (1/aspect) \\ h = cot(ang1/2) \\ aspect = screenWidth/screenHeigth \\ Q = Zf/(Zf - Zn) \end{cases} \quad (3)$$

式(2)是投影变换矩阵，从原点 O 到近剪裁面 A’B’C’D’的最短距离用 Zn 表示，从原点 O 到远剪裁面 ABCD 的最短距离用 Zf 表示，ang1 表示原点到 A’B’中点的夹角值，screenWidth 代表屏幕的宽度，screenHeight 代表屏幕的高度。在经过投影空间变换和视图空间变换后，推导出投影空间中的点 p 和视图空间中的点 v 的关系如下：

$$(p_x * Zn, p_y * Zn, Zn) = (v_x * p_Matrix.m11, v_y * pMatrix.m22, Vz * Q - QZn, v_z) \quad (4)$$

其中 P_x，P_y，P_z 分别表示 p 点的 x，y，z 轴坐标点，而 V_x，V_y，V_z 分别表示 V 的 x，y，z 轴坐标点。由式(5)可得下式：

$$\begin{cases} p_x * Zn = v_x * pMatrix.m11 \\ p_y * Zn = v_y * pMatrix.m22 \\ p_z * Zn = v_z * Q - QZn \end{cases} \quad (5)$$

因为 P 点的 z 值为 0，所以 V 点的各坐标可用式(6)计算得出：

$$\begin{cases} v_x = p_x * Zn / pMatrix.m11 \\ v_y = p_y * Zn / pMatrix.m22 \\ v_z = Zn \end{cases} \quad (6)$$

(3)通过计算得出射线的方向矢量由式(6)得到射线方向矢量的各分量计算公式如下：

$$\begin{cases} v_dir_x = (v_x - 0)/Zn \\ v_dir_y = (v_y - 0)/Zn \\ v_dir_z = (v_z - 0)/Zn \end{cases} \quad (7)$$

(4)最后计算世界坐标下的射线向量，使用矩阵变换，Vworldspace 为待求的世界向量，Vviewspace 为视图空间下的向量，Mview-1 代表视图空间转换到世界空间的矩阵。根据矩阵转换公式就可以得到在世界空间下射线的向量[4]。

$$V_{worldspace} = M_{view}^{-1} * V^{viewspace} \quad (8)$$

在 Three.js 框架内实现鼠标点击监测的过程如图 5 所示：

在实现了鼠标的点击检测，匹配到被点击的图元对象，获得该图元的 ID 标识，通过遍历每个图元所特有的 ID 标识可获取该图元属性信息，利用 HTML5 技术将其打印到屏幕上，实现图元属性查询。实验结果如图 6 所示：

2.3 三维交互

使用 Three.js 中的 createCameraControls 方法创建一个新的 THREE.TrackballControls 实例，传入了

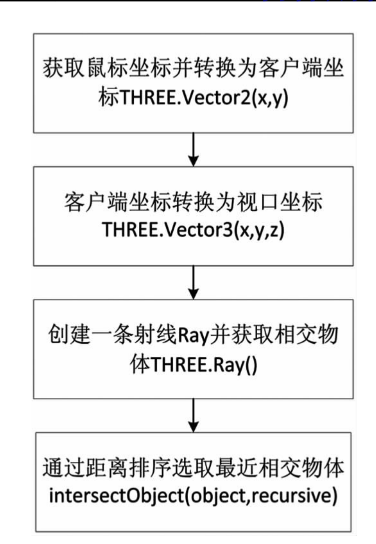


图 5　Three.js 点击检测过程

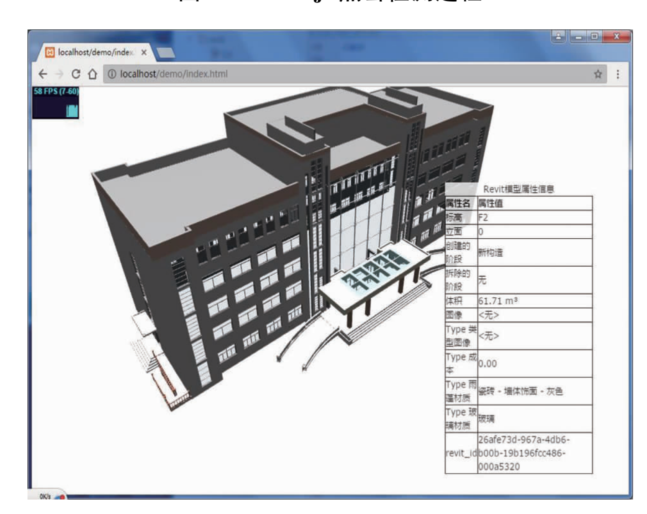


图 6　三维模型可视化单击显示属性信息

一个相机对象和渲染器的 DOM 元素。通过对相机控制的属性设置来控制平移、缩放、旋转等行为。TrackballControls 控制方式如表 1 所示：

表 1　TrackballControls 控制方式

鼠标操作	动作
按住左键、拖动	在场景中转动相机
转动滚轮	放大缩小
按住中键、拖动	放大缩小
按住右键、拖动	在场景中平移

通过轨迹球来实现对相机的操控，同时还需要添加 Web 端对鼠标的监听事件来对鼠标发出的点击、滚轮事件来实现与三维模型的实时交互。根据左右键按下的不同，鼠标事件会用不同的方式来移动相机。通过鼠标操作“放大”“缩小”“平移”“视

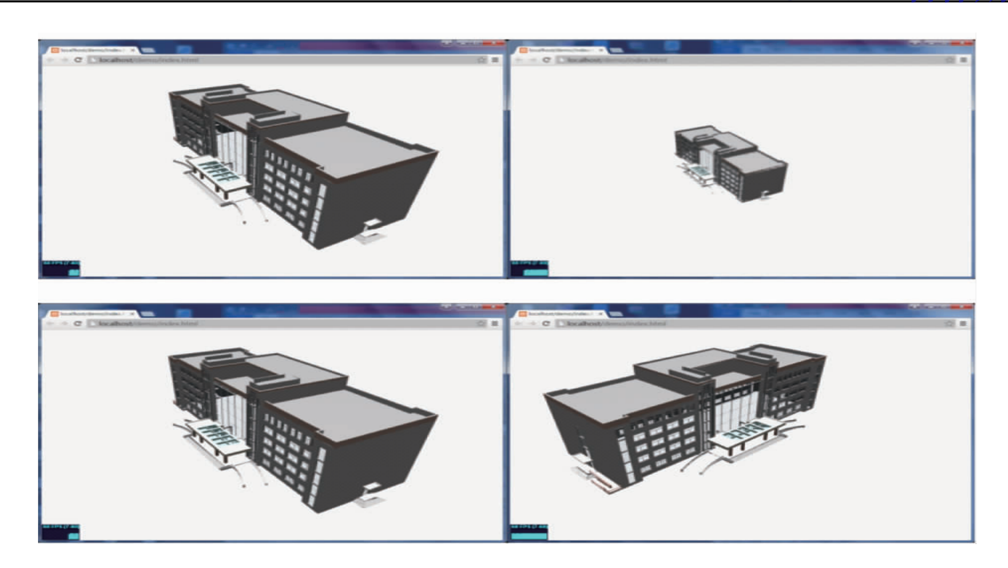


图 7　三维模型缩放旋转效果展示

角调整”就可快捷查看任意关键部位,不会出现卡机不流畅等现象。图 7 分别是对模型旋转、缩放变化图组。

3　结语

本文介绍了 Revit 模型数据导出插件的开发过程,在 VisualStudio2013 平台下通过对 RevitAPI 的调用,实现 Revit 模型的几何数据,属性信息导出为 JSON 格式文件。在此基础上开发了建筑信息模型 Web 端的可视化平台,使用 Three. js 框架完成模型的导入,实现模型属性查询功能,完成模型在 Web 端的三维交互。

参考文献

[1] 何清华, 杨德磊,郑弦. 国外建筑信息模型应用理论与实践现状分析综述[J]. 科技管理研究, 2015(3).

[2] 陈宜. 浅谈 BIM(Revit)的软硬件配置[J]. 建筑创作, 2011(12): 146-151.

[3] 纪博雅, 戚振强. 国内 BIM 技术研究现状[J]. 科技管理研究, 2015(6): 184-190.

[4] 高辰飞, 徐建良. 基于 WebGL 的海洋样品三维可视化研究[J]. 中国科技博览, 2014(3): 428-429.

[5] Autodesk, Inc. Revit2012 API Developer's Guide, 2011.

[6] 郑华, 宿景芳. 面向 Web 的三维模型生成与处理技术[J]. 现代电子技术, 2015(12).

Building Information Model of Web Side Reconstruction and 3D Interaction Method Research

Li Changhua, Zhang Hanyue, Zhou Fangxiao

(*Xi'an University of Architecture and Technology*, *Xi'an* 710055, *China*)

Abstract: Under the present BIM technology background, in view that most existing BIM servers are under C/S frameworks, issues appears in higher request on the computer software and hardware, failure in cross-platform operation, etc. This reports our work on implementing reconstruction of BIM model on web, and realizing collecting the building model and acquiring its properties. The method is mainly divided into two parts. Firstly, the BIM model is derived from Revit, and secondly, the model information exported can achieve 3D interaction on Web, which possesses good user experience.

Key Words: Building Information Model; WebGL; 3D Interaction